Вестник Донского государственного технического университета. ISSN 1992-5980 eISSN 1992-6006. 2018. Т. 18, № 2. С. 214–222
Vestnik of Don State Technical University. ISSN 1992-5980 eISSN 1992-6006 2018. Vol. 18, no. 2, pp. 214-222

ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ
INFORMATION TECHNOLOGY, COMPUTER SCIENCE, AND MANAGEMENT



УДК 004.021.056.55 DOI 10.23947/1992-5980-2018-18-2-214-222

Архитектура и реализация программы онлайн-тестировщика для проверки решений задач по олимпиадному программированию[*]

Е. В. Шагилова[1**]
[1]Национальный исследовательский Мордовский государственный университет имени Н. П. Огарева, г. Саранск, Российская Федерация

Architecture and implementation of online testing to check solutions to Olympiad programming problems[***]

E. V. Shagilova[1**]
[1]Ogarev Mordovia State University, Saransk, Russian Federation

Введение. Рассмотрена возможность автоматизации проверки работ участников олимпиад по программированию. Описана архитектура и работа серверной части системы проверки выполнения олимпиадных задач по программированию.
Материалы и методы. В качестве средств создания приложения были рассмотрены технологии MySQL, PHP, C++, JavaScript, HTML, CSS. Программа-тестировщик реализована на языке C++ для операционных систем семейства Windows NT.
Результаты исследования. Реализована возможность автоматизации проверки работ участников олимпиады в режиме реального времени. Для проверки олимпиадных работ по программированию разработана программа-тестировщик системы.
Обсуждение и заключения. В результате анализа функционирования глобальной сети Internet и технологии «клиент-сервер» были определены возможности для организации взаимодействия приложения с Internet-ресурсами. Разработанное приложение доказало целесообразность и эффективность организации взаимодействия приложения с Internet-ресурсами.

Introduction. The possibility of automated checking the works of participants of the Olympiads in programming is considered. The architecture and operation of the server part of the check system of performing the Olympiad programming tasks is described.
Materials and Methods. The technologies of MySQL, PHP, C ++, JavaScript, HTML, and CSS are considered as the application framework. The test program is implemented in C ++ for operating systems of the Windows NT family.
Research Results. The opportunity of the automated check of the Olympiad participants' works in a real-time mode is fulfilled. A program-tester of the system is developed to check the programming Olympiad works.
Discussion and Conclusions. As a result of the analysis of the global network and client/server technology operation, capabilities for the organization of interaction of the application with Internet-resources are defined. The developed application has proved expediency and efficiency of the interoperability of the application and Internet-resources.

Ключевые слова: онлайн-система, тестировщик, программирование, веб-сервер, интернет, протокол передачи данных.

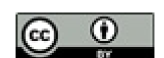
Keywords: online system, tester, programming, web server, Internet, data transfer protocol.

Образец для цитирования. Шагилова, Е. В. Архитектура и реализация онлайн-тестировщика для проверки решений задач по олимпиадному программированию / Е. В. Шагилова // Вестник Дон. гос. техн. ун-та. — 2018. — Т.18, № 2. — С.214–222. DOI: 10.23947/1992-5980-2018-18-2-214-222

For citation: E.V. Shagilova. Architecture and implementation of online testing to check solutions to Olympiad programming problems. Vestnik of DSTU, 2018, vol. 18, no.2, pp. 214–222. DOI: 10.23947/1992-5980-2018-18-2-214-222

Введение. В настоящее время наблюдается тенденция роста количества различных соревнований по программированию. Одна из важных особенностей олимпиад по программированию — возможность автоматизации проверки работ участников. При этом возможна проверка и ранжирование участников в режиме реального времени. В связи с этим появляется необходимость разработки соответствующих решений. Автоматическое тестирование программных решений, обсуждается достаточно широко. На данный момент

[*] Работа выполнена в рамках инициативной НИР.
[**] E-mail:shagilova_elena@mail.ru
[***] The research is done within the frame of the independent R&D.



разработаны и успешно реализуются различные методики автоматического тестирования и программные продукты, основанные на их применении. Исследователи изучают различные аспекты реализации олимпиад по программированию. В ряде публикаций российских ученых [1–6] рассматривались вопросы автоматизированной генерации тестов для олимпиадных задач по программированию, обсуждались и описывались новые подходы к тестированию задач. Поднимаются проблемы совершенствования систем оценивания, основанных на автоматических проверках, а также встает задача разработки многофункционального многопользовательского инструментария [7–12].

Проведенный обзор литературы показал актуальность разработки программы онлайн-тестировщика для проверки решений задач по олимпиадному программированию. Поэтому цель представленной работы — разработка онлайн-системы для проверки решений задач по олимпиадному программированию. Проверка решений осуществляется в серверной части, под которой в данном контексте понимается программа-тестировщик (checker), которая осуществляет непосредственную проверку правильности решенной задачи.

На вход тестировщик получает исходный код проверяемой программы, имя компилятора и номер задачи, а также путь до конфигурационного файла, в котором описаны пути до тестов, ограничения по времени и памяти для проверяемой программы, а также настройки компиляторов. Тестировщик осуществляет компиляцию переданного исходного кода и проверку скомпилированного решения посредством сравнений с заранее подготовленными тестами, замеры времени выполнения и объема использованной памяти, а затем выносит вердикт о правильности переданного решения задачи.

Процесс создания пользовательского интерфейса — очень важный этап в разработке веб-приложения. Большинство создателей программного обеспечения уделяют большое внимание тому, чтобы пользователи чувствовали себя комфортно при работе с их приложениями. Именно поэтому одним из важнейших этапов создания программных средств является разработка и создание пользовательского интерфейса.

В работе рассмотрены основные этапы создания пользовательского интерфейса, серверной части и их взаимодействие внутри системы проверки олимпиадных работ по программированию, а также представлен конечный результат, то есть сама система с интерфейсом.

Материалы и методы

Основными средствами для реализации системы являются HTML, CSS, PHP, MySQL, JavaScript, Apache.

Установка системы начинается с установки веб-сервера. Для работы использовался Apache. Необходимо наличие на сервере установленного пакета Microsoft Visual Studio и FreePascal.

В данной работе реализованы возможности для проверки задач, написанных именно на языках программирования C++ и Pascal.

Программа-тестировщик реализована на языке C++ для операционных систем семейства Windows NT.

Постановка задачи

Необходимо разработать веб-приложение, отвечающее таким требованиям, как высокая нагруженность, интерактивность, низкое потребление ресурсов на стороне клиента, гибкость конфигурации, возможность проверки одной задачи неограниченным количеством тестов, неограниченность количества решений одной задачи, устойчивость к отказу одной из составляющих веб-приложения, географическая независимость места проведения олимпиады.

Высокая нагруженность означает, что система должна сохранять стабильную работку при работе с ней, по крайней мере, 100 человек. Для решения этой задачи применяется кластер какого-либо учебного заведения и правильное проектирование базы данных, а также всех элементов разрабатываемой системы.

Интерактивность предполагает постоянное взаимодействие с пользователем и быстрый отклик на все его действия. В частности, отображается таймер с оставшимся у участника олимпиады временем до окончания олимпиады, выводится подробная информация по каждой выполненной задаче (количество всех тестов и количество правильно пройденных тестов). Для этого используется технология AJAX.

Результаты исследования

Для проведения олимпиады на машинах с любыми техническими характеристиками было выбрано решение, при котором все вычисления происходят на стороне сервера, а клиенту отдается готовый результат всех вычислений. Иными словами, браузер является терминалом, отображающим полученную от сервера информацию.

Для уменьшения ограничений по установке системы была создана директория с конфигурационными файлами, позволяющая настраивать расположение модулей системы по усмотрению администратора, не задевая основную файловую систему сервера.

Также в конфигурационных файлах указывается количество задач и количество тестов и ответов к ним. Таким образом, система «знает» заранее как обрабатывать полученное от клиента решение определенной задачи и из этих же файлов «видит» сколько баллов начислить за каждый тест.

Система может быть доступна из любого места, где есть доступ в интернет. Она разворачивается на веб-сервере и осуществляет работу как обычный веб-сайт. Наглядно принцип работы системы представлен на рис. 1.

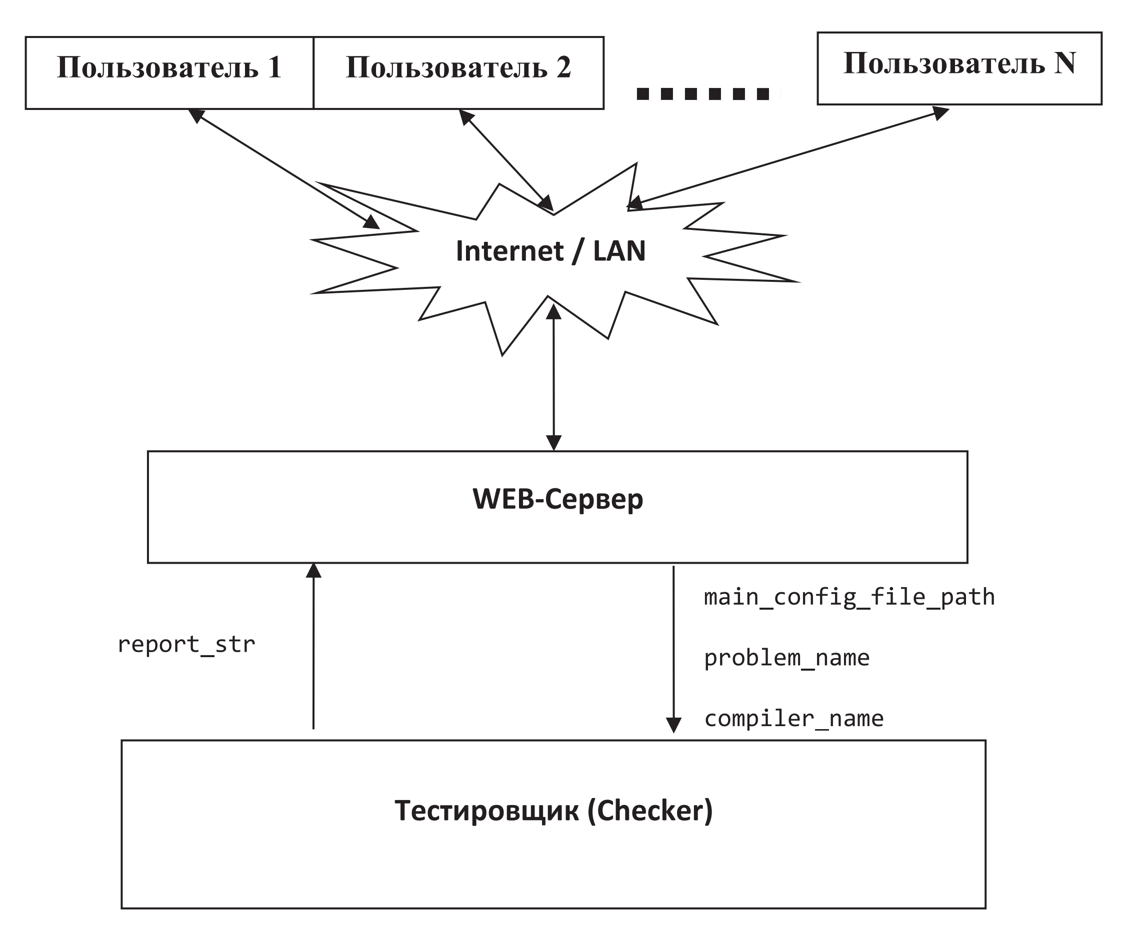


Рис. 1. Архитектура системы для онлайн-проверки решений задач по олимпиадному программированию

Fig. 1. System architecture for online checking solutions to Olympiad programming problems

Для того, чтобы система была устойчива к перебоям с доступом к интернету или электроэнергией, все данные в режиме реального времени вносятся в базу данных. Для этого у каждого из участников олимпиады есть заранее созданная учетная запись (логин и пароль, известный только ему).

Был выбран модульный подход к созданию системы. Это позволяет изменять, дорабатывать или фиксировать проблемы в каждой части системы, независимо от остальных. К примеру, если участник олимпиады присылает программу, в которой содержится код, вызывающий сбой в работе чекера, то прекратит свою работу и выдаст ошибку пользователю только лишь единственный процесс, запущенный для проверки задачи.

Описание чекера

Чекер — это программа, написанная на языке C++, находящаяся на сервере, которая занимается проверкой присланных решений.

Клиент обращается к web-серверу по протоколу прикладного уровня передачи данных HTTP через сеть Internet или LAN. Под клиентом понимается участник олимпиады или администратор web-сервера. Клиент входит в систему. Далее участник олимпиады получает возможность выбрать номер задачи, язык программирования и отправить файл с исходным кодом для проверки, после чего система сообщит результат. Если клиент является администратором, то он имеет возможность посмотреть список участников олимпиады, зарегистрировать новых участников, а также посмотреть ранжированные результаты олимпиады. Web-сервер

взаимодействует с программой тестировщик (checker) следующим образом: через аргументы командной строки тестировщику передаются следующие параметры в указанном порядке:

— main_config_file_path — путь до файла конфигураций, в котором описаны пути до тестов, ограничения по времени и памяти для проверяемой программы, а также настройки компиляторов. Это строка, в которой следует избегать пробелов.

— problem_name — имя или порядковый номер задачи. Это должна быть строка, содержащая только символы десятичных цифр, латинские буквы нижнего, латинские буквы верхнего регистра.

— compiler_name — имя компилятора. В настоящий момент поддерживаются компиляторы C++ и Pascal, поэтому это может быть строка: «pas» или «cpp».

— profile_path — путь до директории, в которой хранятся файлы текущего тестируемого пользователя по текущей задаче. Рекомендуется для каждого тестируемого пользователя создавать отдельную директорию, а внутри — отдельную поддиректорию для тестируемой задачи. Это строка, в которой следует избегать пробелов.

— source_name — имя файла тестируемого исходного кода. Это строка, содержащая имя файла тестируемого исходного кода. Файл исходного кода должен быть помещен внутри директории profile_path.

Таким образом, аргументы командной строки могут представлять, например, такую строку:

«C:\\checker\\main_config.txt 1 cpp C:\\checker\\users\\vitaly\\1 main.cpp».

Тестировщик, в свою очередь, формирует отчет по отправленной задаче:

— report_str — отчет по отправленной задаче — строка, выводимая программой тестировщик в стандартный поток вывода.

Грамматика report_str (для описания грамматики будет использоваться расширенная форма Бэкуса-Наура, разработанная Никлаусом Виртом):

```
report_str = "report" eol verdict.
verdict = compilation_error | (intervals ";" balls eol {test_id report eol}).
interval = integer " " | integer " - " integer " ".
intervals = interval {interval}.
balls = integer.
test_id = integer.
report = correct | incorrect | crash | tle | mle.
correct = "correct " time " " memory.
time = integer.
memory = integer.
incorrect = "incorrect".
crash = "crash".
tle = "tle".
mle = "mle".
compilation_error = "compilation_error" eol.
```

Пример отчета программы тестировщика по отправленной задаче:

```
report
1 - 2 5 7 - 10 ;60
1 correct 100 1806336
2 correct 105 1806336
3 mle
4 crash
5 correct 130 1807336
6 tle
7 correct 130 1807336
8 correct 130 1807336
9 correct 125 1807336
10 correct 125 1807336
```

В первой строке слово report говорит о начале отчета тестирующей программы по отправленному решению. Во второй строке указано, какие тесты пройдены и какие балы начислены за пройденные тесты. В конфигурационном файле указывается количество балов, которое соответствует каждому тесту, в случае, если он будет пройден. В последующих строках дана детальная информация по пройденным тестам. В случае, если тест пройден, то программа выводит correct, а также время и память, которая понадобилась проверяемой программе. Если превышено ограничение по памяти, то программа выводит mle. Если превышено ограничение по времени, то программа выводит tle. Если во время исполнения тестируемой программы произошел сбой, например, деление на ноль либо какой-то другой, приведший к аварийному завершению программы, то в отчете будет crash. Если не удалось скомпилировать полученный исходный код программы, то тестировщик выводит compilation_error, в этом случае информация о тестах выводиться не будет.

Из вышеописанной архитектуры можно сделать следующий вывод. Поскольку программа тестировщик является отдельным независимым компонентом системы в целом и по результатам своей работы она сообщает максимально полную информацию о тестируемом решении, то web-сервер может разрабатываться для большого разнообразия систем правил проведения олимпиад по программированию. Этот тестировщик подойдет как для правил ACM ICPC, так и для школьных олимпиад.

Программа checker состоит из четырех основных частей: заголовочного файла reader.h с его реализацией в reader.cpp — содержит функционал для чтения конфигурационных файлов и тестов, а также структуры хранения считанной информации; builder.h — содержит функционал для синтаксического анализа конфигурационных файлов, в которых описано, как использовать компиляторы, а также функционал, который отвечает за компиляцию переданных системе исходных файлов; executor.h — содержит функционал, отвечающий за запуск скомпилированных решений, мониторинг их работы по количеству использованной памяти и затраченного времени, а также структуры, описывающие результаты их работы; matcher.h — содержит функционал, отвечающий за сравнение правильных ответов для теста и тех, что вернула проверяемая программа.

Интерфейс системы

Интерфейс системы проверки олимпиадных задач должен быть максимально простым для того, чтобы не отвлекать участников от самих заданий и быть интуитивно понятным. По этой причине было принято решение отказаться от всех привычных элементов дизайна любого сайта и оставить только то, что необходимо для работы пользователя с системой.

Взаимодействие начинается с процесса авторизации пользователя в системе (рис. 2). Перед началом олимпиады подготавливается база данных, в которой есть все логины и пароли, которые раздаются участникам.

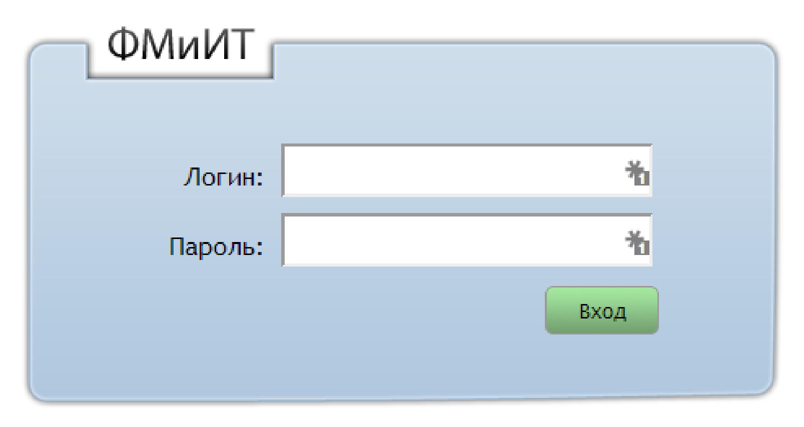


Рис. 2. Форма авторизации пользователя

Fig. 2. User authorization form

При правильном заполнении авторизационных данных появляется основной интерфейс системы (рис. 3). В нем содержится информация о количестве всех задач и времени до окончании олимпиады. Имеется возможность отключить ограничение по времени администратором.

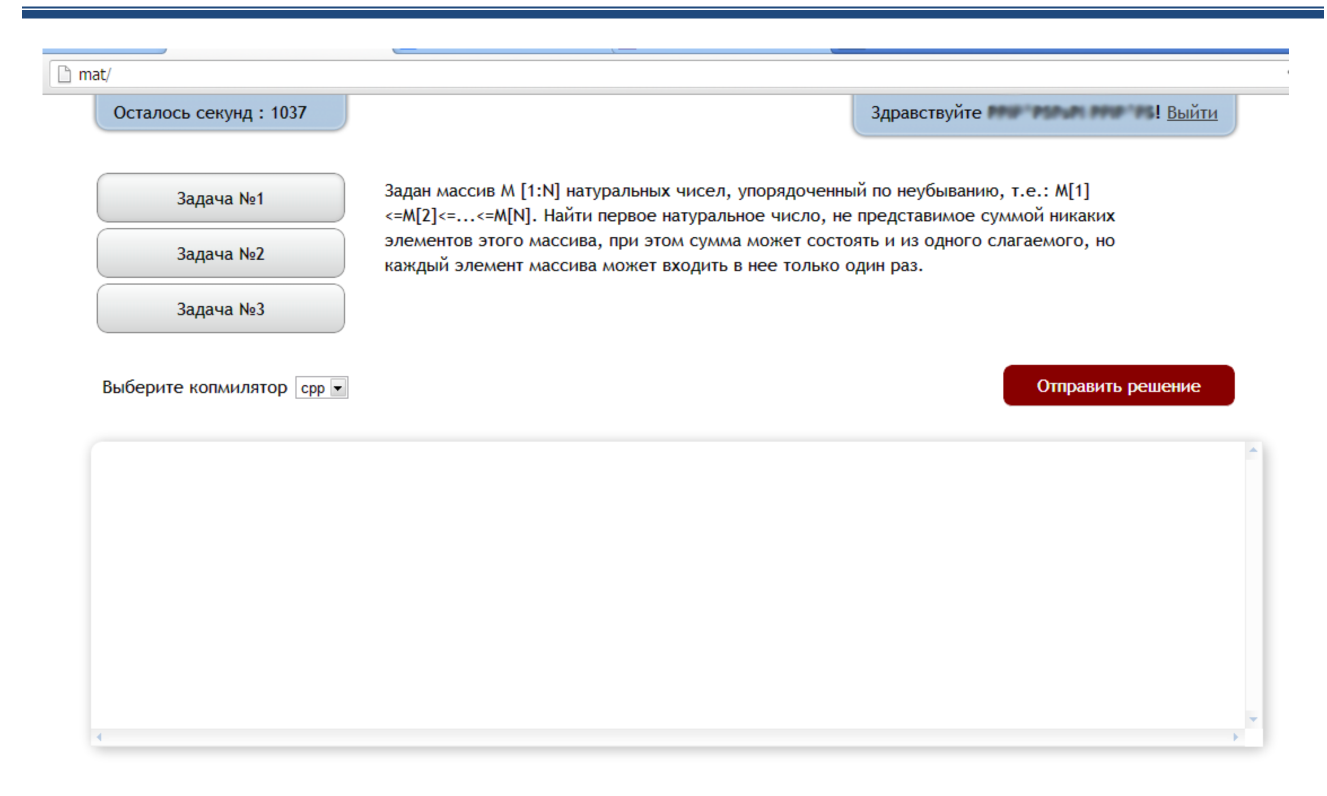


Рис. 3. Интерфейс пользователя

Fig. 3. User interface

При клике на кнопки выбора задач в левой части происходит обновление блока с текстом задачи без перезагрузки всего интерфейса. Когда пользователь посчитает, что готов отослать решение на сервер, то он должен проверить правильно ли выбран компилятор. Это делается с помощью выпадающего списка, находящегося под блоком выбора задач. Затем нужно нажать на кнопку «Отправить решение». Это самый яркий элемент интерфейса системы, так как выделен красным цветом.

В самом низу находится поле, отображающее информацию по каждой задаче: общей сумме набранных баллов, общем количестве тестов и информация о правильности выполнения каждого из тестов.

По окончании выполнения олимпиады участник может завершить работу с системой с помощью кнопки «Выйти», находящуюся в правом верхнем углу интерфейса. Если произойдет попытка отправить решение после момента, когда закончится время, то система выдаст соответствующее предупреждение и обработка задачи не произойдет.

В мире веб-приложений существует проблема совместимости технологий, которыми описывается интерфейс, и программных средств, пытающихся его отобразить. Интерфейс, представленный в данной работе корректно отображается в браузерах Internet Explorer выше 8 версии, Firefox выше 15 версии, всех версиях Google Chrome и Opera Next, а также всех браузерах, построенных на ядре Chromium, например Яндекс браузер, Amigo от mail.ru и других.

Работа администратора с системой начинается с аналогичного окна авторизации, представленного на рис. 2. Затем появляется форма регистрации новых участников в системе (рис. 4). После заполнения нужных полей следует нажать на зеленую кнопку «Зарегистрировать» и участник окажется внесенным в базу данных. Этот способ добавления участников довольно медленный. Ниже описан более автоматизированный способ заполнения базы данных новыми пользователями.

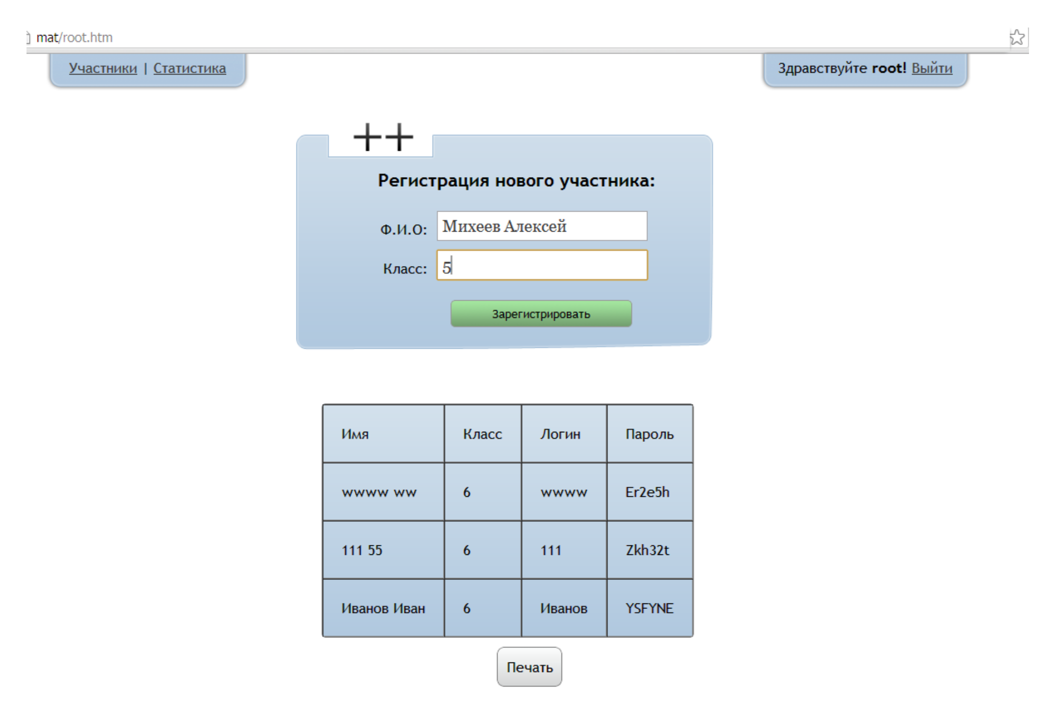


Рис. 4. Интерфейс администратора системы

Fig. 4. System administrator interface

После заполнения администратором базы данных появляется возможность распечатать данные об участниках и использовать их как раздаточный материал. Во время проведения самой олимпиады можно, нажав на кнопку «статистика», следить за успехами соревнующихся участников. После окончания олимпиады, нажав на эту же кнопку, будет предоставлена вся информация о результатах олимпиады в ранжированном по убыванию виде. Можно будет увидеть, кто и сколько тестов выполнил в каждой конкретной задаче и распечатать ее при необходимости.

По окончанию работы с веб-приложением администратору можно также, как и обычному пользователю, завершить сессию работы с системой с помощью кнопки «Выйти», находящейся в правом верхнем углу.

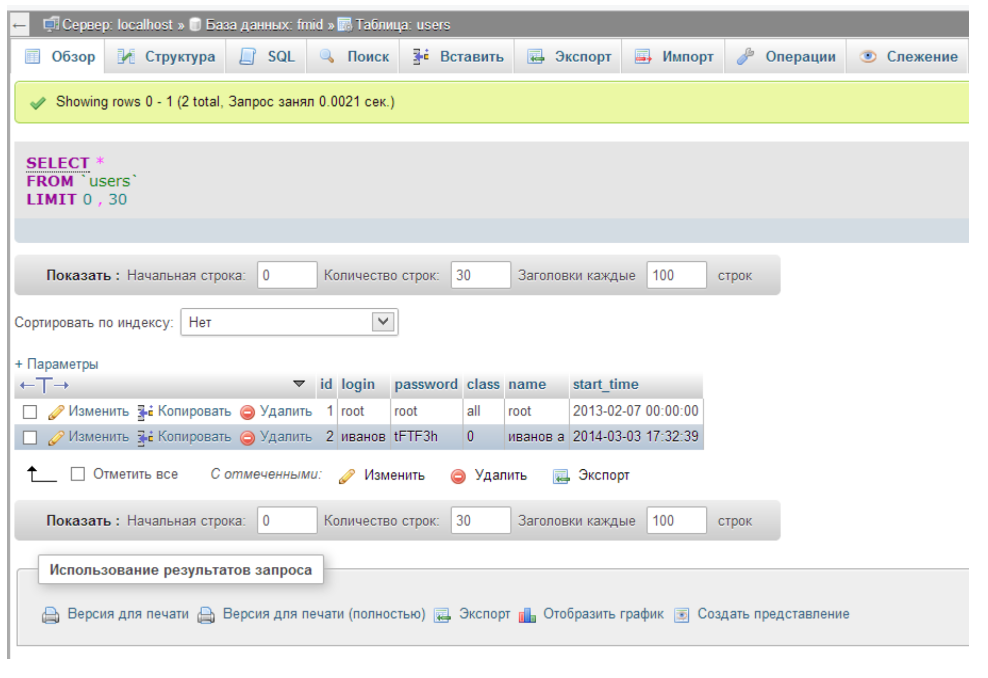


Рис. 5. Интерфейс PhpMyAdmin

Fig. 5. PhpMyAdmin interface

Для быстрого заполнения базы данными об участниках можно воспользоваться веб-приложением PhpMyAdmin (рис. 5). Обычно он является заранее установленным на веб-серверах, поэтому сложностей в его поисках возникать не должно. Чтобы автоматизировать процесс, необходимо составить SQL запрос, который сгенерирует все необходимые данные, за исключением логинов. Затем в интерфейсе, представленном на рис. 4, необходимо перейти на вкладку «SQL» и ввести запрос для таблицы «users».

Обсуждение и заключение

В данной работе рассмотрен процесс создания системы для проверки олимпиадных работ по программированию, этапы создания пользовательского интерфейса, серверной части и их взаимодействие внутри приложения.

Тестировать решения задач по программированию умеют многие сервисы, но все они имеют свои недостатки. Разрабатывая систему, представленную в этой работе, были приложены все усилия, чтобы избавиться от них. Она способна проверять правильность решения задач, представляемых Министерством образования РФ, а также подстраиваться под нужды школ, университетов и факультетов, для которых такие олимпиады являются неотъемлемой частью процесса обучения. Все основные типы контроля знаний (варианты ответа, множественный выбор, ответ-слово, ответ-текст, многовариантность задачи), все основные типы контроля правильности выполнения в системе есть.

Система устанавливается на сервере и при наличии внешнего IP у университета или школы способна принимать решения олимпиадных задач по протоколу http. Предусмотрена гибкая система назначения оценок, выстраивание рейтинга участников.

Имеется возможность проводить олимпиады одновременно в нескольких классах, давая каждому классу отдельный набор задач. В дополнение к этому можно ограничить время проведения олимпиады для каждого класса отдельно.

Администратор системы (организатор олимпиады) может в режиме онлайн видеть ход олимпиады, всех участников, а также знать, кто и сколько выполнил задач и сколько тестов в каждой из присланных задач.

В результате анализа функционирования глобальной сети Internet и технологии «клиент-сервер» были определены возможности для организации взаимодействия приложения с Internet-ресурсами.

Разработанное приложение доказало целесообразность и эффективность организации взаимодействия приложения с Internet-ресурсами. Тестировщик системы для проверки олимпиадных работ по программированию окажет значительную пользу как участникам, так и организаторам олимпиад.

Библиографический список

1. Буздалов, М. В. Генерация тестов для олимпиадных задач по программированию с использованием генетических алгоритмов / М. В. Буздалов // Научно-технический вестник Санкт-Петербургского университета информационных технологий, механики и оптики. — 2011. — № 2. — С. 72–77.

2. Корнеев, Г. А. Автоматическое тестирование решений на соревнованиях по программированию / Г. А. Корнеев, Р. А. Елизаров // Телекоммуникации и информатизация образования. — 2003. — № 1. — С. 61–73.

3. Назаренко, А. С. Аспекты автоматизации тестирования при проведении олимпиад по программированию / А. С. Назаренко, В. О. Скрипачёв // Информационные технологии в обеспечении федеральных государственных образовательных стандартов. — 2014. — № 3. — С. 147–151.

4. Евстропов, Г. О. Системы оценивания в задачах с автоматической проверкой на олимпиадах по программированию / Г. О. Евстропов // Информатика и образование. — 2016. — № 3 (272). — С. 65–67.

5. Макиева, З. Д. Проектирование автоматизированной системы проверки олимпиадных заданий по программированию / З. Д. Макиева // Известия Кыргызского государственного технического университета им. И. Раззакова. — 2016. — Т. 38. — С. 54–61.

6. Иванова, С. А. Построение сервиса автоматизированной проверки решений задач по информатике «Информатик-ассистент» / С. А. Иванова, В. В. Иванов, Н. В. Николаева // Инновационные технологии в науке и образовании. — 2015. — № 4 (4). — С. 150–154.

7. Применение автоматизированной системы тестирования в учебном процессе / Е. Н. Боженкова [и.др.] // Новые информационные технологии в образовании (НИТО-Байкал). — 2010. — № 2. — С. 161–163.

8. Сверчкова, Г. В. Автоматизированная система проверки результатов олимпиады по программированию / Г. В. Сверчкова, Д. И. Кислицын // Сборник статей студ., аспирантов и магистр. «Информационные системы и технологии». — 2016. — С. 30–34.

9. Рогачева, Е. В. Опыт использования систем автоматизированной проверки решений при обучении программированию / Е. В. Рогачева // Высшая школа. — 2015. — № 9. — С. 55–58.

10. Самощенко, Ю. Ю. Исследование эффективности автоматизированной проверки решений при проведении олимпиад по программированию / Ю. Ю. Самощенко // Молодой ученый. — 2016. — № 11. — С. 223–226.

11. Автоматизированная система тестирования программ / В. А. Соловьев [и др.] // Электронные средства и системы управления. — 2012. — № 1. — С. 188–191.

12. Автоматизированная система тестирования программ / С. А. Черепанов [и.др.] // Электронные средства и системы управления. — 2014. — № 2. — С. 61–65.

References

1. Buzdalov, M.V. Generatsiya testov dlya olimpiadnykh zadach po programmirovaniyu s ispol'zovaniem geneticheskikh algoritmov. [Tests generation for Olympiad programming tasks using genetic algoritms.] Scientific and Technical Journal of Information Technologies, Mechanics and Optics, 2011, no. 2, pp. 72–77 (in Russian).

2. Korneev, G.A., Yelizarov, E.A. Avtomaticheskoe testirovanie resheniy na sorevnovaniyakh po programmirovaniyu. [Automatic testing of solutions on programming competitions.] Telekommunikatsii i informatizatsiya obrazovaniya, 2003, no. 1, pp. 61–73 (in Russian).

3. Nazarenko, A.S., Skripachev, V.I. Aspekty avtomatizatsii testirovaniya pri provedenii olimpiad po programmirovaniyu. [Aspects of testing automation during the programming Olympiads.] Informatsionnye tekhnologii v obespechenii federal'nykh gosudarstvennykh obrazovatel'nykh standartov. [Information technologies in providing Federal state educational standards.] 2014, no. 3, pp. 147–151 (in Russian).

4. Yevstropov, G.O. Sistemy otsenivaniya v zadachakh s avtomaticheskoy proverkoy na olimpiadakh po programmirovaniyu [Assessment systems in problems with automatic testing in Olympiads on programming.] Informatics and Education, 2016, no. 3 (272), pp. 65–67 (in Russian).

5. Makieva, Z.D. Proektirovanie avtomatizirovannoy sistemy proverki olimpiadnykh zadaniy po programmirovaniyu. [Development of the automated verification system for programming Olympiad tasks.] Journal Kyrgyz State Technical University after I. Razzakov , 2016, vol. 38, pp. 54–61 (in Russian).

6. Ivanova, S.A., Ivanov, V.V., Nikolaeva, N.V. Postroenie servisa avtomatizirovannoy proverki resheniy zadach po informatike «Informatik-assistent» [Construction of service for automated verification of solutions to computer science problems “Information scientist-Assistant”.] Innovatsionnye tekhnologii v nauke i obrazovanii, 2015, no. 4 (4), pp. 150–154 (in Russian).

7. Bozhenkova, E.N., et al. Primenenie avtomatizirovannoy sistemy testirovaniya v uchebnom protsesse. [Application of the automated testing system in the educational process.] Novye informatsionnye tekhnologii v obrazovanii (NITO-Baykal). [New information technologies in education (NITO-Baikal).] 2010, no. 2, pp. 161–163 (in Russian).

8. Sverchkova, G.V., Kislitsyn, D.I. Avtomatizirovannaya sistema proverki rezul'tatov olimpiady po programmirovaniyu. [Automated system for checking results of programming Olympiad.] Информационные системы и технологии. — 2016. — № 3. — С. 30–34 (in Russian).

9. Rogacheva, E.V. Opyt ispol'zovaniya sistem avtomatizirovannoy proverki resheniy pri obuchenii programmirovaniyu. [Experience in the use of automated solutions testing systems for programming education.] Vysshaya shkola, 2015, no. 9, pp. 55–58 (in Russian).

10. Samoshchenko, Y.Y. Issledovanie effektivnosti avtomatizirovannoy proverki resheniy pri provedenii olimpiad po programmirovaniyu. [Study on automated verification efficiency of solutions during programming Olympiads.] Young Scientist, 2016, no. 11, pp. 223–226 (in Russian).

11. Solovyev, A., et al. Avtomatizirovannaya sistema testirovaniya programm. [Automated system for testing programs.] Elektronnye sredstva i sistemy upravleniya. [Electronic means and control systems.] 2012, no.1, pp. 188–191 (in Russian).

12. Cherepanov, S.A., et al. Avtomatizirovannaya sistema testirovaniya programm. [Automated system for testing programs.] Elektronnye sredstva i sistemy upravleniya. [Electronic means and control systems.] 2014, no. 2, pp. 61–65 (in Russian).

Поступила в редакцию 14.03.2018
Сдана в редакцию 18.03.2018
Запланирована в номер 15.05.2018

Received 14.03.2018
Submitted 18.03.2018
Scheduled in the issue 15.05.2018

Об авторе:

Шагилова Елена Викторовна,
доцент кафедры фундаментальной информатики, Мордовского государственного университета им. Н. П. Огарева (430005, Россия, г. Саранск, ул. Большевистская, д. 68), кандидат педагогических наук, доцент,
ORCID: https://orcid.org/0000-0003-0267-6082
shagilova_elena@mail.ru

Author:

Shagilova, Elena V.
associate professor of the Fundamental Informatics Department, Ogarev Mordovia State University (68, Bolshevistskaya St., Saransk, 430005, Russia), Cand.Sci. (Pedagogics), associate professor,
ORCID: https://orcid.org/0000-0003-0267-6082
shagilova_elena@mail.ru